

Interpretable Two-level Boolean Rule Learning for Classification

Guolong Su^{*} Dennis Wei[†] Kush R. Varshney[†] Dmitry M. Malioutov[†]

Abstract

This paper proposes algorithms for learning two-level Boolean rules in Conjunctive Normal Form (CNF, i.e. AND-of-ORs) or Disjunctive Normal Form (DNF, i.e. OR-of-ANDs) as a type of human-interpretable classification model, aiming for a favorable trade-off between the classification accuracy and the simplicity of the rule. Two formulations are proposed. The first is an integer program whose objective function is a combination of the total number of errors and the total number of features used in the rule. We generalize a previously proposed linear programming (LP) relaxation from one-level to two-level rules. The second formulation replaces the 0-1 classification error with the Hamming distance from the current two-level rule to the closest rule that correctly classifies a sample. Based on this second formulation, block coordinate descent and alternating minimization algorithms are developed. Experiments show that the two-level rules can yield noticeably better performance than one-level rules due to their dramatically larger modeling capacity, and the two algorithms based on the Hamming distance formulation are generally superior to the other two-level rule learning methods in our comparison. A proposed approach to binarize any fractional values in the optimal solutions of LP relaxations is also shown to be effective.

Keywords

Interpretable Classifier, Linear Programming Relaxation

1 Introduction

Boolean rules are an important classification model for machine learning and data mining. A typical Boolean rule connects a subset of binary input features with the logical operators conjunction (“AND”), disjunction (“OR”), and negation (“NOT”) to form the prediction. As an example, a Boolean rule in [8] for the prediction of 10 year coronary heart disease (CHD) risk for a 45

year old male can be expressed as follows:

```
IF      1. NOT smoke; OR
        2. total cholesterol < 160; AND
        systolic blood pressure < 140;
THEN (10 year CHD risk < 5%)=TRUE.
```

This is a two-level rule in DNF (OR-of-ANDs), where in the lower level, conjunctions of binary features build *clauses* and in the upper level, the disjunction of the clauses forms the predictor.

An advantage of Boolean rules is high human interpretability [3, 4]. The features included in the learned rule provide key reasons behind the prediction results; in the example above, not smoking may be the reason for the prediction of low 10 year CHD risk. These reasons can be easily understood by the users.

Human interpretability has high importance in a wide range of applications such as medicine and business [4, 8], where results from prediction models are generally presented to a human decision maker/agent who makes the final decision. Such a decision maker often needs an understanding of the reasons for the prediction before accepting the result; thus, high prediction accuracy without providing the reasons is not sufficient for the model to be trusted. As an example, medical diagnosis models [8] may predict a high risk of certain diseases for a patient; a doctor then needs to know the underlying factors to compare with his/her domain knowledge, take the correct action, and communicate with the patient. Another application requiring interpretability is fraud detection [6], where convincing reasons are needed to justify further auditing.

This paper considers learning two-level Boolean rules from datasets, with the joint criteria of both classification accuracy and human interpretability measured by the total number of features used (i.e. sparsity) [3]. Two optimization-based formulations are introduced. The objective function in the first formulation is a weighted combination of the total number of classification errors and the sparsity, based on which we extend a previously proposed LP relaxation approach from one-level to two-level rules. The second formulation replaces the 0-1 classification error cost with the Hamming distance from the current rule to the closest rule that correctly classifies a sample; we propose

^{*}Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA, USA. (email: guolong@mit.edu)

[†]Mobile, Solutions, and Mathematical Sciences Department, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. (email: dwei, krvarshn, dmalioutov@us.ibm.com)

block coordinate descent and alternating minimization approaches for optimizing the objective in the second formulation. To tackle the issue of fractional optimal solutions to LP relaxations, we introduce a new binarization method to convert LP solutions into binary values. Experiments show that compared with one-level rules, the two-level rules can have noticeably lower error rate as well as more flexible accuracy-simplicity trade-offs. The two algorithms based on the Hamming distance formulation generally have superior performance among the approaches for two-level rule learning that we compare, and the new binarization method is shown to be effective.

The remainder of this paper is organized as follows. Section 2 reviews related work and fields. After the problem formulations in Section 3, optimization approaches are introduced in Section 4 and evaluated in Section 5. Section 6 concludes this work.

2 Review of Existing Work

The two-level Boolean rules in this work are examples of sparse decision rule lists, one of the major classes of interpretable models [4]. Decision trees constitute another class that can represent the same Boolean functions and be converted to decision rule lists [14], although they may differ in the representation complexity depending on the dataset [4]. Section 2.1 and 2.2 focus on existing work in learning one-level and two-level Boolean rules, respectively. The one-level rule learning method in [10] forms a building block in the current work.

2.1 One-level Rule Learning in [10] A standard binary supervised classification problem is considered in [10]. We have a training dataset with n labeled samples; the i^{th} sample has a binary label $y_i \in \{0, 1\}$ and in total d binary features $a_{i,j} \in \{0, 1\}$ ($1 \leq j \leq d$). The goal is to learn a classifier $\hat{y}(\cdot)$ that can generalize well to unseen feature vectors sampled from the same distribution as the training dataset.

The class of classifiers considered in [10] consists of one-level Boolean rules, which take only a conjunction (or disjunction) of selected features. De Morgan's laws show an equivalence between corresponding conjunctive and disjunctive rules

$$y = \bigwedge_{j \in C} x_j \Leftrightarrow \bar{y} = \bigvee_{j \in C} \bar{x}_j,$$

where \bar{y} and \bar{x}_j mean the negation of binary variables y and x_j , respectively. Due to this equivalence, algorithms in [10] focus on the disjunctive rule

$$\hat{y}_i = \bigvee_{j=1}^d a_{i,j} w_j, \text{ for } 1 \leq i \leq n,$$

where binary decision variable $w_j \in \{0, 1\}$ indicates whether the j^{th} feature is selected in the rule, and $\hat{y}_i \in \{0, 1\}$ is the prediction of the i^{th} sample.

Replacing binary operators with linear-algebraic expressions, a mixed integer program is formulated for the one-level rule learning problem [10]:

$$(2.1) \quad \min_{w_j} \sum_{i=1}^n \xi_i + \theta \cdot \sum_{j=1}^d w_j$$

$$(2.2)$$

$$\text{s.t. } \xi_i = \max \left\{ 0, \left(1 - \sum_{j=1}^d a_{i,j} w_j \right) \right\}, \text{ for } y_i = 1,$$

$$(2.3) \quad \xi_i = \sum_{j=1}^d a_{i,j} w_j, \text{ for } y_i = 0,$$

$$(2.4) \quad w_j \in \{0, 1\}, \text{ for } 1 \leq j \leq d.$$

The objective function is a combination of accuracy and sparsity with the balance controlled by the parameter θ . The accuracy related costs ξ_i for false negatives and false positives are formulated in (2.2) and (2.3), respectively.

Relaxation of (2.4) into $0 \leq w_j \leq 1$ yields a linear program that is efficiently solvable [10]. Sufficient conditions for the relaxation to be exact are discussed in [10].

2.2 Two-level Rule Learning Two-level Boolean rules have significantly larger modeling capacity than one-level rules. In fact, if we include the negations of input features, then two-level rules can represent any Boolean function of the input features [12, 13], which does not hold for one-level rules.

Two algorithms are proposed in [10] for rule set learning, based on the one-level learning algorithm. The first algorithm uses the set covering approach [11] and obtains a two-level rule. Suppose we want to learn a rule in DNF (OR-of-ANDs). After training the first clause with the entire training set, we remove the samples with output 1 from the first clause; the predictions on these samples have been determined regardless of the other clauses. Then we train the second clause with the remaining samples, and repeat this remove-train procedure for the rest of clauses. Since this set covering approach is a one-pass greedy-style algorithm, there should be space for improvement. The second algorithm for rule sets in [10] applies boosting, in which the predictor is a *weighted combination* of rules rather than a two-level rule and thus hinders interpretability.

Another algorithm for DNF learning is the Hamming Clustering (HC) approach [13], which uses greedy methods to iteratively cluster samples in the same category and with features close to each other in Hamming

distance. HC may be seen as bottom-up, whereas our algorithms are top-down and treat the training dataset more globally. Experiments in [13] seem to imply HC produces a high number of clauses, which hinders interpretability.

There are a number of other methods and fields related to two-level rule learning. First, Bayesian approaches in [8, 16] typically utilize approximate inference algorithms to obtain the MAP solution or to produce posterior distribution over decision lists. However, the assignment of prior and likelihood in the Bayesian framework may not always be clear, and certain approximate inference algorithms may have high computational cost. Second, Logical Analysis of Data (LAD) [2] learns *patterns* for both positive and negative samples by techniques such as set covering [11], and typically builds a classifier by a weighted combination of the patterns, i.e. not a two-level rule. Third, learnability of Boolean formulae is considered in [7] from the perspective of probably approximately correct (PAC) learning. Different from our problem, the setup of [7] and related work typically assumes positive or negative samples can be generated on demand and without noise. Fourth, two-level logic optimization in circuit design [12] considers simplifying two-level rules that exactly match a given truth table. However, in rule learning, it is neither needed nor desirable to exactly match a noisy dataset.

3 Problem Formulation

The goal is to learn a two-level Boolean rule in the Conjunctive Normal Form (AND-of-ORs) from a training dataset¹, with the same setup of binary supervised classification as in Section 2.1. In the lower level of the rule, we form each clause by the disjunction of a selected subset of input features; in the upper level, the final predictor is formed by the conjunction of all clauses. Suppose the total number of clauses is fixed and denoted by R . If we let the binary decision variables $w_{j,r}$ represent whether to include the j^{th} feature in the r^{th} clause, then the output of the r^{th} clause for the i^{th} sample is

$$(3.5) \quad \hat{v}_{i,r} = \bigvee_{j=1}^d (a_{i,j} w_{j,r}), \text{ for } 1 \leq i \leq n, 1 \leq r \leq R.$$

Then, the predictor \hat{y}_i satisfies

$$(3.6) \quad \hat{y}_i = \bigwedge_{r=1}^R \hat{v}_{i,r}, \text{ for } 1 \leq i \leq n.$$

Although this setup has a fixed R , an option to “disable” a clause can be introduced to reduce the total number

of actual clauses if the assigned R is too large. For a CNF rule, a clause can be regarded as disabled if its output is always 1. Thus, we can pad the input feature matrix with a trivial “always true” feature $a_{i,0} = 1$ for all samples, and also include the corresponding decision variables $w_{0,r}$ for all clauses. The sparsity cost for $w_{0,r}$ can be lower than other variables or even zero. If $w_{0,r} = 1$, then the r^{th} clause has output 1 and is thus disabled in the CNF rule. This option might reduce accuracy with the tradeoff of improved sparsity.

In certain cases, DNF rules could be more natural than CNF. A CNF learning algorithm can apply to DNF learning by De Morgan’s laws:

$$y = \bigvee_{r=1}^R \left(\bigwedge_{j \in \mathcal{C}_r} x_j \right) \Leftrightarrow \bar{y} = \bigwedge_{r=1}^R \left(\bigvee_{j \in \mathcal{C}_r} \bar{x}_j \right)$$

where \mathcal{C}_r is the index set of features selected in the r^{th} clause. To learn a DNF rule with a CNF learning algorithm, we can first negate both features and labels of all samples, then learn a CNF rule with the negated features and labels, and finally use the decision variables $w_{j,r}$ with the original features to construct a DNF rule. Thus, Sections 3 and 4 focus on CNF only.

Two formulations are introduced with different accuracy costs in Section 3.1 and 3.2, respectively.

3.1 Formulation with 0-1 Error Cost A natural choice of the accuracy-related cost is the total number of misclassifications (i.e. 0-1 error for each sample). With the sparsity cost as the sum of the number of features used in each clause, a formulation is as below

$$(3.7) \quad \min_{w_{j,r}} \sum_{i=1}^n |\hat{y}_i - y_i| + \theta \cdot \sum_{r=1}^R \sum_{j=1}^d w_{j,r}$$

$$(3.8) \quad \text{s.t. } \hat{y}_i = \bigwedge_{r=1}^R \left(\bigvee_{j=1}^d (a_{i,j} w_{j,r}) \right), \text{ for } 1 \leq i \leq n, \\ w_{j,r} \in \{0, 1\}, \text{ for } 1 \leq j \leq d, 1 \leq r \leq R.$$

There are a few challenges in this formulation. First, the same as one-level rule learning, the two-level rule learning problem is combinatorial. Second, all the clauses are symmetric in (3.7) and (3.8), however, we generally would like the clauses to be distinct since duplication of clauses is inefficient.

3.2 Formulation with Minimal Hamming Distance This formulation has the two motivations below. First, it is potentially desirable to have a finer-grained accuracy cost than the 0-1 error in (3.7). As an example, consider two CNF rules, both with two clauses, predicting the same sample with ground truth label $y_i = 1$.

¹We assume the negation of each feature is included as another input feature; if not, we can pad the input features with negations.

Suppose both clauses in the first rule predict 0, while only one clause in the second rule predicts 0 and the other predicts 1. Although both rules misclassify this sample after taking “AND” of their two clauses, the second rule is closer to correct than the first one. If we use an iterative algorithm to refine the learned rule, it might be beneficial for the accuracy cost term to favor the second rule in this example, which could push the solution towards being correct. The second motivation for this formulation is to avoid identical clauses by training each clause with a different subset of samples, as done in [10, 11, 15].

In the new formulation, the accuracy cost for a single sample is the *minimal Hamming distance* from a given CNF rule to an *ideal* CNF rule, where the latter means a rule that correctly classifies this sample. The Hamming distance between two CNF rules is the total number of $w_{j,r}$ that are different in the two rules. An intuitive explanation of this minimal Hamming distance is the smallest number of modifications (i.e. negations) of the current rule $w_{j,r}$ that are needed to correct a misclassification on a sample, i.e. how far is the rule from being correct.

For mathematical formulation, we introduce *ideal clause outputs* $v_{i,r}$ with $1 \leq i \leq n$ and $1 \leq r \leq R$ to represent a CNF rule that correctly classifies the i^{th} sample. The values of $v_{i,r}$ are always consistent with the ground truth labels, i.e. $y_i = \bigwedge_{r=1}^R v_{i,r}$ for all $1 \leq i \leq n$. We let $v_{i,r}$ have a ternary alphabet $\{0, 1, \text{DC}\}$, where $v_{i,r} = \text{DC}$ means that we “don’t care” about the value of $v_{i,r}$. With this setup, if $y_i = 1$, then $v_{i,r} = 1$ for all $1 \leq r \leq R$; if $y_i = 0$, then $v_{i,r_0} = 0$ for at least one value of r_0 , and we can have $v_{i,r} = \text{DC}$ for all $r \neq r_0$. In implementation, $v_{i,r} = \text{DC}$ implies the removal of the i^{th} sample in the training or updating for the r^{th} clause, which generally leads to a different training subset for each clause.

Denote η_i as the minimal Hamming distance from the current CNF rule $w_{j,r}$ to an ideal CNF rule for the i^{th} sample. We derive η_i for positive and negative samples, respectively. Since $y_i = 1$ implies $v_{i,r} = 1$ for all r , for each clause with output 0 in the current rule, at least one positive feature needs to be included to match $v_{i,r} = 1$. Thus, the minimal Hamming distance for a positive sample is the number of clauses with output 0:

$$\eta_i = \sum_{r=1}^R \max \left\{ 0, \left(1 - \sum_{j=1}^d a_{i,j} w_{j,r} \right) \right\}, \text{ for } y_i = 1.$$

For $y_i = 0$, we first consider for fixed r the minimal Hamming distance between the r^{th} clauses of the current rule and an ideal rule where $v_{i,r} = 0$. We need to negate $w_{j,r}$ in the current rule for j with

$w_{j,r} = a_{i,j} = 1$ to match $v_{i,r} = 0$, and thus the minimal Hamming distance of this clause is $\sum_{j=1}^d a_{i,j} w_{j,r}$. Then, since $v_{i,r} = 0$ needs to hold for at least one value of r while all other $v_{i,r}$ can be DC, the minimal Hamming distance of the CNF rule is given by the minimum over r , i.e. setting $v_{i,r_0} = 0$ with

$$(3.9) \quad r_0 = \arg \min_{1 \leq r \leq R} \left(\sum_{j=1}^d a_{i,j} w_{j,r} \right).$$

Combining all analysis above, the new formulation with the minimal Hamming distance cost is as below

$$(3.10) \quad \min_{w_{j,r}} \sum_{i=1}^n \eta_i + \theta \cdot \sum_{r=1}^R \sum_{j=1}^d w_{j,r}$$

$$\text{s.t. } \eta_i = \sum_{r=1}^R \max \left\{ 0, \left(1 - \sum_{j=1}^d a_{i,j} w_{j,r} \right) \right\}, \text{ for } y_i = 1,$$

$$(3.11) \quad \eta_i = \min_{1 \leq r \leq R} \left(\sum_{j=1}^d a_{i,j} w_{j,r} \right), \text{ for } y_i = 0,$$

$$w_{j,r} \in \{0, 1\}, \text{ for } 1 \leq j \leq d, 1 \leq r \leq R.$$

The binary variables $w_{j,r}$ can be further relaxed to $0 \leq w_{j,r} \leq 1$. The minimum over r in (3.11) implies the continuous relaxation of (3.10) is generally non-convex with $R > 1$, making the exact solution challenging.

Letting $R = 1$ in formulation (3.10), we can see it becomes identical to formulation (2.1) in one-level rule learning [10]. Thus, the accuracy cost in (2.1) could be interpreted as the minimal Hamming distance.

To simplify description of algorithms later, we show a formulation (3.12) below, which is equivalent to (3.10) but involves both $v_{i,r}$ and $w_{j,r}$. Taking the minimization over $v_{i,r}$ in (3.12) with fixed $w_{j,r}$ eliminates the variables $v_{i,r}$, and (3.12) becomes identical to (3.10).

$$(3.12)$$

$$\min_{w_{j,r}, v_{i,r}} \sum_{i=1}^n \sum_{r=1}^R \left[\mathbb{1}_{v_{i,r}=1} \cdot \max \left\{ 0, \left(1 - \sum_{j=1}^d a_{i,j} w_{j,r} \right) \right\} + \mathbb{1}_{v_{i,r}=0} \cdot \sum_{j=1}^d a_{i,j} w_{j,r} \right] + \theta \cdot \sum_{r=1}^R \sum_{j=1}^d w_{j,r}$$

$$(3.13) \quad \text{s.t. } \bigwedge_{r=1}^R v_{i,r} = y_i, \text{ for } 1 \leq i \leq n, \\ v_{i,r} \in \{0, 1, \text{DC}\}, \text{ for } 1 \leq i \leq n, 1 \leq r \leq R, \\ w_{j,r} \in \{0, 1\}, \text{ for } 1 \leq j \leq d, 1 \leq r \leq R.$$

4 Optimization Approaches

This section discusses various optimization approaches to the two-level rule learning problem. Based on the for-

mulation in Section 3.1, we generalize the LP approach from one-level rule learning to two-level rules by proper relaxation in Section 4.1. Based on the formulation in Section 3.2, we propose the block coordinate descent algorithm in Section 4.2 and the alternating minimization algorithm in Section 4.3 for the objective (3.12). Since all algorithms utilize LP relaxations, Section 4.4 considers the binarization problem if the result of the LP is not binary.

4.1 Two-level Linear Programming Relaxation

This approach considers the 0-1 error formulation (3.7) and directly generalizes the idea of replacing binary operations “AND” and “OR” with linear-algebraic operations, as used in one-level rule learning [10]. Since “AND” and “OR” are defined only on binary points, there are various interpolations of these functions on fractional points, and thus both convex and concave interpolations exist for both operators. The “OR” function has the following interpolations [5]

$$\bigvee_{j=1}^d x_j = \max_{1 \leq j \leq d} \{x_j\} = \min \left\{ 1, \sum_{j=1}^d x_j \right\},$$

where the first is convex and the second is concave, both of which are the respective tightest interpolations.

The logical “AND” operator also has the tightest convex and concave interpolations as [5]

$$\bigwedge_{j=1}^d x_j = \max \left\{ 0, \left(\sum_{j=1}^d x_j \right) - (d-1) \right\} = \min_{1 \leq j \leq d} \{x_j\}.$$

Since the predictor \hat{y}_i of the two-level rule in (3.8) is a composition of “AND” and “OR” operators, it is possible to properly interpolate it using both a convex function and a concave function by composing the individual interpolations of the two operators. From (3.5) and (3.6), a convex interpolation of \hat{y}_i is

$$\hat{y}_i = \max \left\{ 0, \left(\sum_{r=1}^R \max_{1 \leq j \leq d} \{a_{i,j} w_{j,r}\} \right) - (R-1) \right\},$$

and a concave interpolation can be obtained similarly.

Denote the 0-1 error cost for the i^{th} sample as ψ_i . Since the errors ψ_i in (3.7) should be minimized, if $y_i = 1$, then $\psi_i = 1 - \hat{y}_i$ and thus we need the concave interpolation for \hat{y}_i ; if $y_i = 0$, then $\psi_i = \hat{y}_i$ and thus the convex interpolation is needed. Finally, the formulation in (3.7) can be exactly converted into a mixed integer

program as follows:

$$(4.14) \quad \begin{aligned} & \min_{w_{j,r}, \psi_i, \beta_{i,r}} \sum_{i=1}^n \psi_i + \theta \cdot \sum_{r=1}^R \sum_{j=1}^d w_{j,r} \\ & \text{s.t. } \psi_i \geq 0, \forall i, \\ & \psi_i \geq 1 - \sum_{j=1}^d a_{i,j} w_{j,r}, \text{ for } y_i = 1, \forall r, \\ & \psi_i \geq \left(\sum_{r=1}^R \beta_{i,r} \right) - (R-1), \text{ for } y_i = 0, \\ & \beta_{i,r} \geq a_{i,j} w_{j,r}, \text{ for } y_i = 0, \forall j, \forall r, \\ & w_{j,r} \in \{0, 1\}, \forall j, \forall r. \end{aligned}$$

If we relax the decision variables $w_{j,r}$ to the interval $[0, 1]$, then we have a linear program.

Unfortunately, numerical results seem to suggest that this LP relaxation is likely to have fractional values in the optimal solution $w_{j,r}$, and the optimal ψ_i may possibly be all close to 0, which may be undesirable since ψ_i aims to represent the 0-1 error cost term. A possible reason is that the gap between the convex and concave interpolations may loosen the LP and enable fractional results with lower cost than binary solutions.

4.2 Block Coordinate Descent Algorithm

This algorithm considers the decision variables in a single clause ($w_{j,r}$ with a fixed r) as a block of coordinates, and performs block coordinate descent to minimize the Hamming distance objective function in (3.12). Each iteration updates a single clause with all the other $(R-1)$ clauses fixed, using the one-level rule learning algorithm [10]. We denote r_0 as the clause to be updated.

The optimization of (3.12) even with $(R-1)$ clauses fixed still involves a joint minimization over w_{j,r_0} and the ideal clause outputs $v_{i,r}$ for $y_i = 0$ ($v_{i,r} = 1$ for $y_i = 1$ and thus fixed), so the exact solution could still be challenging. To simplify, we fix the values of $v_{i,r}$ for $y_i = 0$ and $r \neq r_0$ to the actual clause outputs $\hat{v}_{i,r}$ in (3.5) with the current $w_{j,r}$ ($r \neq r_0$). Now we assign v_{i,r_0} for $y_i = 0$: if there exists $v_{i,r} = \hat{v}_{i,r} = 0$ with $r \neq r_0$, then this sample is guaranteed to be correctly classified and we can assign $v_{i,r_0} = \text{DC}$ to minimize the objective in (3.12); in contrast, if $\hat{v}_{i,r} = 1$ holds for all $r \neq r_0$, then the constraint (3.13) requires $v_{i,r_0} = 0$.

This derivation leads to the updating process as follows. To update the r_0^{th} clause, we remove all samples that have label $y_i = 0$ and are already predicted as 0 by at least one of the other $(R-1)$ clauses, and then update the r_0^{th} clause with the remaining samples using the one-level rule learning algorithm [10].

There are different choices of which clause to update in an iteration. For example, we can update clauses cyclically or randomly, or we can try the update for each clause and then greedily choose the one with the minimum cost. The greedy update is used in our experiments.

The initialization of $w_{j,r}$ in this algorithm also has different choices. For example, one option is the set covering method [10], as is used in our experiments. Random or all-zero initialization can also be used.

4.3 Alternating Minimization Algorithm This algorithm uses the Hamming distance formulation (3.12) and alternately minimizes with respect to the decision variables $w_{j,r}$ and the ideal clause outputs $v_{i,r}$. Each iteration has two steps: update $v_{i,r}$ with the current $w_{j,r}$, and update $w_{j,r}$ with the new $v_{i,r}$. The latter step is simpler and will be first discussed.

With fixed values of $v_{i,r}$, the minimization over $w_{j,r}$ is relatively straight-forward: the objective in (3.12) becomes separated into R terms, each of which depends only on a single clause $w_{j,r}$ with a fixed r . Thus, all clauses are decoupled in the minimization over $w_{j,r}$, and the problem becomes parallel learning of R one-level clauses. Explicitly, the update of the r^{th} clause will first remove all the samples with $v_{i,r} = \text{DC}$, and then utilize the one-level rule learning algorithm [10].

The update over $v_{i,r}$ with fixed $w_{j,r}$ follows the discussion in Section 3.2: for positive samples $y_i = 1$, $v_{i,r} = 1$, and for the negative samples $y_i = 0$, $v_{i,r_0} = 0$ for r_0 defined in (3.9) and $v_{i,r} = \text{DC}$ for $r \neq r_0$. For negative samples with a “tie”, i.e. non-unique r_0 in (3.9), tie breaking is achieved by a “clustering” approach similar to the spirit of [13]. First, for each clause $1 \leq r_0 \leq R$, we compute its cluster center in the feature space by taking the average of $a_{i,j}$ (for each j) over samples i for which r_0 is minimal in (3.9) (including ties). Then, each sample with a tie is assigned to the clause with the closest cluster center in ℓ_1 -norm among all minimal r_0 in (3.9).

Similar to the block coordinate descent algorithm, various options exist for initializing $w_{j,r}$ in this algorithm. The set covering approach [10] is used in our experiments.

4.4 Redundancy Aware Binarization This section discusses a solution to a potential issue with the LP relaxation that is widely used in the algorithms proposed in this paper. Although there are conditions under which the optimal solution to the LP relaxation for one-level rule learning is guaranteed to be binary [10], we are not aware of similar guarantees in two-level rule learning; in addition, these conditions are unlikely to

always hold with a real-world and noisy dataset. Thus, the optimal solution to LP may have fractional values, in which case we need to convert them into binary. If LP already yields a binary optimal solution, then the binarization methods here will not change it.

A straight-forward binarization method is to compare each $w_{j,r}$ from LP with a specified threshold, as done in [10]. However, empirical results seem to suggest that the resulting binarized rule may have *redundancy*, making the rule unnecessarily complex and possibly influencing the accuracy.

The following improved binarization method considers three types of *redundancy* sets of binary features in a single *disjunctive* clause. Among the features in each redundancy sets, no more than one feature will appear in any single clause of the optimal CNF rule².

The first type of redundancy set corresponds to *nested* features. If binary features a_{i,j_1} and a_{i,j_2} satisfy $a_{i,j_1} \leq a_{i,j_2}$ for all samples, then these two features cannot both appear in a single clause in the optimal CNF rule; otherwise, since $a_{i,j_1} \vee a_{i,j_2} = a_{i,j_2}$, removing a_{i,j_1} from the clause keeps the same output and improves the sparsity, leading to a better rule. If there is a nested set $a_{i,j_1} \leq a_{i,j_2} \leq \dots \leq a_{i,j_P}$ ($\forall 1 \leq i \leq n$), then at most one feature from this set can be selected in a single clause in the optimal CNF rule.

The second type consists of complementary binary features, when we have the option to “disable” a clause as explained in Section 3. Since complementary features a_{i,j_1} and a_{i,j_2} satisfy $a_{i,j_1} \vee a_{i,j_2} = 1$ ($\forall i$), the optimal CNF rule cannot have both of them in a single clause, otherwise disabling this clause by $w_{0,r} = 1$ and $w_{j,r} = 0$ ($j > 0$) keeps the output and improves sparsity.

The third type also applies only when we have the option to disable a clause. This type can happen with two nested sets that are pairwise complementary, especially if some binary features are obtained by thresholding continuous valued features. For example, suppose we have six binary features from thresholding the same continuous feature c_i with thresholds $\tau_1 < \tau_2 < \tau_3$:

$$\begin{aligned} a_{i,1} &= (c_i \leq \tau_1), \quad a_{i,2} = (c_i \leq \tau_2), \quad a_{i,3} = (c_i \leq \tau_3), \\ a_{i,4} &= (c_i > \tau_1), \quad a_{i,5} = (c_i > \tau_2), \quad a_{i,6} = (c_i > \tau_3). \end{aligned}$$

The “zigzag” path $(a_{i,4}, a_{i,5}, a_{i,2}, a_{i,3})$ forms a redundancy set, since at most one out of the four features can be selected in a fixed clause of the optimal CNF rule, otherwise either the first or the second redundancies above will happen and thus the rule is not optimal. There are typically multiple “zigzag” paths, e.g.

²This statement holds for both formulations (3.7) and (3.12); for simplicity, we will focus on the 0-1 error cost formulation for illustration.

$(a_{i,4}, a_{i,1}, a_{i,2}, a_{i,3})$ and $(a_{i,4}, a_{i,5}, a_{i,6}, a_{i,3})$.

The new binarization approach takes the above types of redundancies into account. For illustration, suppose all binary features are obtained by thresholding continuous valued features. In a clause and for a fixed continuous valued feature, we sweep over all non-redundant combinations of the binary features induced by this continuous feature and obtain the one with minimal cost. Since the total number of non-redundant combinations for nested and zigzag features is linear and quadratic with the number of thresholds, respectively, the sweeping is relatively efficient with a single continuous feature. However, joint minimization across all continuous features seems combinatorial and challenging. Thus, we first sort continuous features in the decreasing order by the sum of corresponding decision variables in the optimal solution to the LP relaxation, and then sequentially binarize the decision variables induced by each continuous feature.

5 Numerical Evaluation

5.1 Setup This section evaluates the algorithms with UCI repository datasets [9], including connectionist bench sonar (Sonar), BUPA liver disorders (Liver), Pima Indian diabetes (Pima), and Parkinsons (Parkin). The continuous valued features in these datasets are converted to binary using quantile thresholds.

The goal is to learn a DNF rule (OR-of-ANDs) from each dataset. We use stratified 10-fold cross validation, and then average the test and training error rates over these 10 folds. All LPs are solved by CPLEX version 12 [1]. The sparsity parameter $\theta = A \times 10^B$ where we sweep $A = 1, 2, 5$ and $B = -4, -3, -2, -1, 0, 1$, for a total of 18 values. We sweep the total number of clauses in the DNF rule between $R = 1$ and $R = 5$; the option to “disable” a clause (which can reduce R) is *not* used in the evaluation, except in Section 5.5 where we compare the results with/without such an option.

Algorithms in comparison and their abbreviations are: two-level LP relaxation (TLP), block coordinate descent (BCD), alternating minimization (AM), set covering [10, 11] (SCS: simple binarization with threshold at 0.2, SCN: new redundancy-aware binarization), decision list [15] in IBM SPSS (DL), and decision trees [14] (C5.0: C5.0 with rule set option in IBM SPSS, CART: classification and regression trees algorithm in Matlab’s classregtree function). TLP, BCD, and AM all use the redundancy-aware binarization. The maximum number of iterations in BCD and AM is set as 100.

We show results on both the minimal average test error rate obtained from the 18 different values of θ and the Pareto front for the tradeoff between accuracy and sparsity.

Table 1: Minimal Average Test Error Rate (unit: %)

Dataset	R	SCS	SCN	TLP	BCD	AM	DL	C5.0	CART
Sonar	1	30.3	25.5	25.5	25.5	25.5	38.5	25.0	28.4
	2	29.8	23.6	28.4	25.5	21.2			
	3	26.0	22.1	30.8	25.5	24.0			
	4	25.5	23.6	29.8	20.2	22.6			
	5	28.4	23.6	29.8	23.6	18.3			
Liver	1	42.0	42.0	39.7	42.0	42.0	45.2	36.5	37.7
	2	40.6	40.9	41.4	35.7	34.2			
	3	40.3	41.4	40.9	36.8	33.6			
	4	39.7	40.6	42.0	37.1	34.2			
	5	39.4	40.6	41.7	33.9	33.0			
Pima	1	26.6	26.7	25.4	26.7	26.7	31.4	24.9	28.9
	2	26.7	26.2	28.0	24.9	22.7			
	3	27.5	26.4	26.7	25.7	23.7			
	4	27.5	27.1	26.3	25.7	24.1			
	5	27.5	27.1	25.1	25.7	24.9			
Parkin	1	15.9	15.9	14.4	15.9	15.9	25.1	16.4	12.8
	2	13.8	13.8	14.9	12.8	14.4			
	3	14.4	14.4	14.4	12.3	13.8			
	4	14.4	14.4	14.9	13.3	14.9			
	5	14.4	14.4	14.9	13.3	15.4			

5.2 Minimal Average Test Error Rate The minimal average test error rates achieved among the 18 values of θ for all algorithms are listed in Table 1, where R denotes the total number of clauses. The results for DL, C5.0, and CART are cited from [10]. We refer the reader to [10] for results from other classifiers that are generally not interpretable; the accuracy of our algorithms is generally quite competitive with them.

For each algorithm and each dataset, the number marked with bold font is the lowest error rate among $1 \leq R \leq 5$. There are a few observations from these results. First, most bold-font numbers appear in rows with $R > 1$. Since $R = 1$ corresponds to the one-level rules while $R > 1$ corresponds to two-level rules, the two-level rules can reduce error rate on these datasets, especially significant for block coordinate descent and alternating minimization algorithms. Second, the block coordinate descent and alternating minimization algorithms generally have superior performance to the other methods for two-level rule learning in our comparison; however, the two-level LP relaxation does not seem to have as good performance. Thus, we focus on block coordinate descent and alternating minimization algorithms in the remainder of this section. Third, for the Sonar dataset with the same R , the set covering approach with new binarization has noticeably lower error rates than with simple binarization, which shows the effectiveness of the redundancy-aware binarization.

Fourth, we can see that for a fixed dataset and a fixed algorithm, the error rate does not decrease monotonically with R , indicating overfitting with too many clauses.

As a preliminary comparison with the Hamming Clustering approach [13], we consider “Pima” which is the only dataset shared by this work, [10], and [13]. HC has 25.0% test error rate with an average of 85 features used in the rule as reported in [13], while block coordinate descent and alternating minimization algorithms have lower minimal error rates of 24.9% (average of 6.3 features used) and 22.7% (average of 6 features used) when $R = 2$, respectively. Thus, our algorithms on Pima dataset produce rules with higher accuracy and significantly fewer features used³.

5.3 Pareto Fronts with Different Numbers of Clauses The Pareto fronts with different numbers of clauses are shown in Fig. 1, where we vary R from 1 to 5. Fig. 1 (a) and (b) show the average test and training error rates of the alternating minimization algorithm on the Pima dataset, while Fig. 1 (c) and (d) show the error rates of the block coordinate descent algorithm on the Liver dataset. Each point in the figure corresponds to the pair of average error rate and the average number of features in the learned DNF rule that is obtained at one of the 18 values of θ , and the Pareto fronts are denoted by lines for ease of visualization.

The following observations are implied by Fig. 1. First, a comparison of the Pareto fronts of $R = 1$ and $R > 1$ suggests that two-level rules may have more flexible tradeoff between accuracy and simplicity. Second, as shown in Fig. 1 (b) and (d), with the increase of R , the learned rule typically uses more features and has lower training error rates. However, the exact tendency of the Pareto front of the test error rate may depend on the complexity of datasets: in Fig. 1 (a), the Pareto front becomes worse with the increase of R when $R > 2$, implying overfitting on this relatively simple dataset; in contrast, for the relatively complex Liver dataset in Fig. 1 (c), the minimum test error rate has a decrease at $R = 5$ with more features used, which seems to suggest that $R = 5$ does not overfit yet.

5.4 Pareto Fronts of Different Algorithms The Pareto fronts of the average test error rates for different algorithms on the Sonar and Liver datasets with $R = 5$ are shown in Fig. 2 (a) and (b), respectively. Comparing the block coordinate descent and alternating minimization algorithms, we can see that when the total number of features used is very small, the block

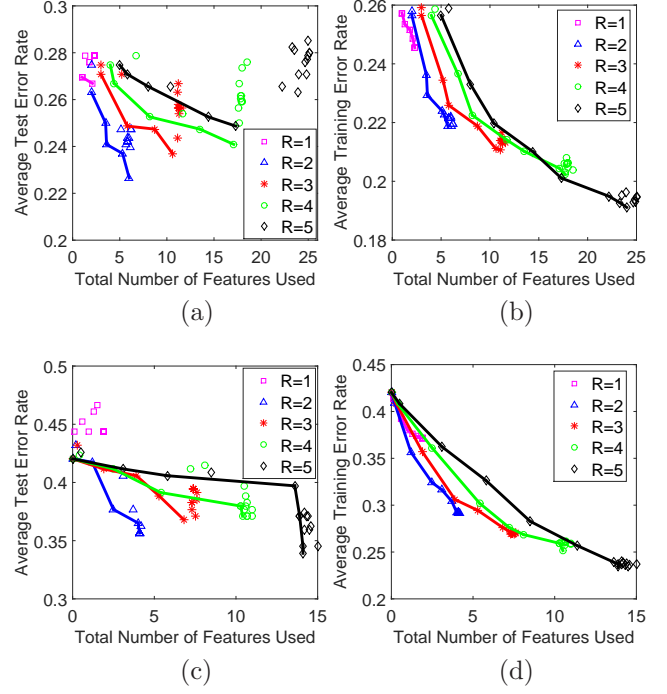


Figure 1: Comparison of Pareto Fronts with Different Numbers of Clauses: (a) Pima AM Test Error Rate, (b) Pima AM Training Error Rate, (c) Liver BCD Test Error Rate, (d) Liver BCD Training Error Rate.

coordinate descent algorithm typically has lower error rate; however, when the total number of features used increases, the alternating minimization algorithm may start to outperform. Comparing the set covering approach with the simple and new binarization, the new binarization generally obtains sparser rules with improved or similar accuracy.

5.5 Pareto Fronts with/without the Option to Disable a Clause Fig. 3 shows the comparison of Pareto fronts of the average test error rates with and without the option to “disable” a clause by an “always true” feature. This option generally improves sparsity, while the error rate may remain similar or increase.

6 Conclusion

This paper has provided two optimization-based formulations for two-level Boolean rule learning, the first based on 0-1 classification error and the second on Hamming distance. Three algorithms have been developed, namely the two-level LP relaxation, block coordinate descent, and alternating minimization. A redundancy-aware binarization method has been introduced.

Numerical results show that two-level Boolean

³There are two differences in setup: HC uses 12-fold cross validation [13], while we use 10-fold; the parameters to convert continuous features into binary may potentially be different.

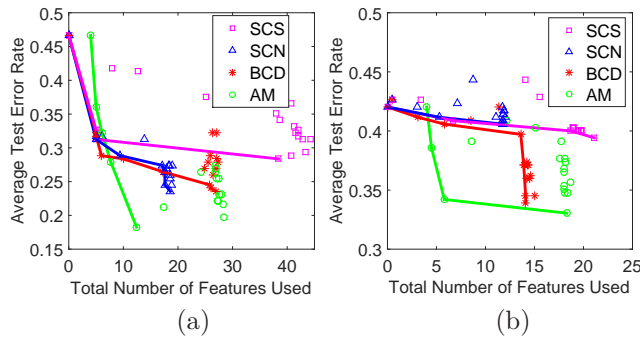


Figure 2: Comparison of Pareto Fronts with Different Algorithms: (a) Sonar $R = 5$, (b) Liver $R = 5$.

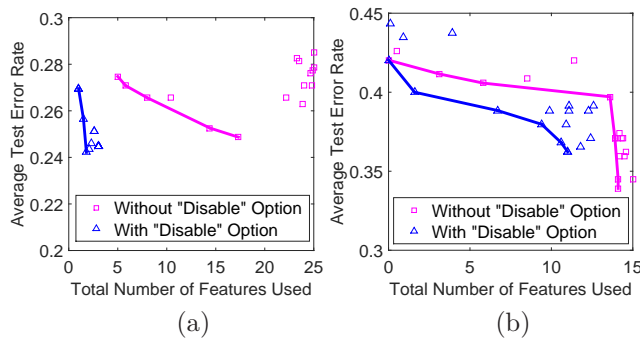


Figure 3: Comparison of Pareto Fronts with and without the Option to Disable a Clause: (a) Pima AM $R = 5$, (b) Liver BCD $R = 5$.

rules have noticeably lower error rate and more flexible accuracy-simplicity tradeoffs on certain complex datasets than one-level rules. However, too many clauses may cause overfitting, and the optimal number of clauses may depend on the complexity of the dataset.

The block coordinate descent and alternating minimization algorithms can work with noisy datasets and generally outperform the other methods for two-level rule learning in our comparison. For the tradeoff between accuracy and simplicity, block coordinate descent algorithm may dominate alternating minimization when we require the total number of features used to be very small; in contrast, alternating minimization algorithm may outperform with more features used.

The new redundancy-aware binarization has been shown more effective than simple thresholding binarization in certain situations.

Acknowledgment

The authors thank for V. S. Iyengar, A. Mojsilović, K. N. Ramamurthy, and E. van den Berg for conversations

and support. The authors are thankful for the assistance in experiments by using datasets from [9].

References

- [1] IBM ILOG CPLEX optimization studio. <http://www-03.ibm.com/software/products/en/ibmilogcpleoptstud>.
- [2] E. BOROS, P. L. HAMMER, T. IBARAKI, A. KOGAN, E. MAYORAZ, AND I. MUCHNIK, *An implementation of logical analysis of data*, IEEE Trans. Knowl. Data Eng., 12 (2000), pp. 292–306.
- [3] J. FELDMAN, *Minimization of Boolean complexity in human concept learning*, Nature, 407 (2000), pp. 630–633.
- [4] A. A. FREITAS, *Comprehensible classification models – a position paper*, ACM SIGKDD Explor., 15 (2014), pp. 1–10.
- [5] R. HÄHNLE, *Proof theory of many-valued logic-linear optimization-logic design: Connections and interactions*, Soft Comput., 1 (1997), pp. 107–119.
- [6] V. S. IYENGAR, K. B. HERMIZ, AND R. NATARAJAN, *Computer-aided auditing of prescription drug claims*, Health Care Manag. Sci., 17 (2014), pp. 203–214.
- [7] M. KEARNS, M. LI, L. PITT, AND L. VALIANT, *On the learnability of Boolean formulae*, in Proc. Annu. ACM Symp. on Theory of Comput., 1987, pp. 285–295.
- [8] B. LETHAM, C. RUDIN, T. H. MCCORMICK, AND D. MADIGAN, *Building interpretable classifiers with rules using Bayesian analysis*, Department of Stat. Tech. Report tr609, Univ. of Washington, (2012).
- [9] M. LICHMAN, *UCI machine learning repository*. <http://archive.ics.uci.edu/ml>, Univ. of Calif., Irvine, School of Information and Computer Sciences, 2013.
- [10] D. M. MALIOUTOV AND K. R. VARSHNEY, *Exact rule learning via Boolean compressed sensing*, in Proc. Int. Conf. Mach. Learn., 2013, pp. 765–773.
- [11] M. MARCHAND AND J. SHAW-TAYLOR, *The set covering machine*, J. Mach. Learn. Res., 3 (2002), pp. 723–746.
- [12] P. C. MCGEER, J. V. SANGHAVI, R. K. BRAYTON, AND A. L. SANGIOVANNI-VINCENTELLI, *ESPRESSO-SIGNATURE: A new exact minimizer for logic functions*, IEEE Trans. VLSI Syst., 1 (1993), pp. 432–440.
- [13] M. MUSELLI AND D. LIBERATI, *Binary rule generation via Hamming Clustering*, IEEE Trans. Knowl. Data Eng., 14 (2002), pp. 1258–1268.
- [14] J. R. QUINLAN, *Simplifying decision trees*, Int. J. Man-Mach. Studies, 27 (1987), pp. 221–234.
- [15] R. L. RIVEST, *Learning decision lists*, Mach. Learn., 2 (1987), pp. 229–246.
- [16] T. WANG, C. RUDIN, F. DOSHI-VELEZ, Y. LIU, E. KLAMPFL, AND P. MACNEILLE, *Bayesian Or's of And's for interpretable classification with application to context aware recommender systems*, tech. report, MIT, 2015. Submitted.